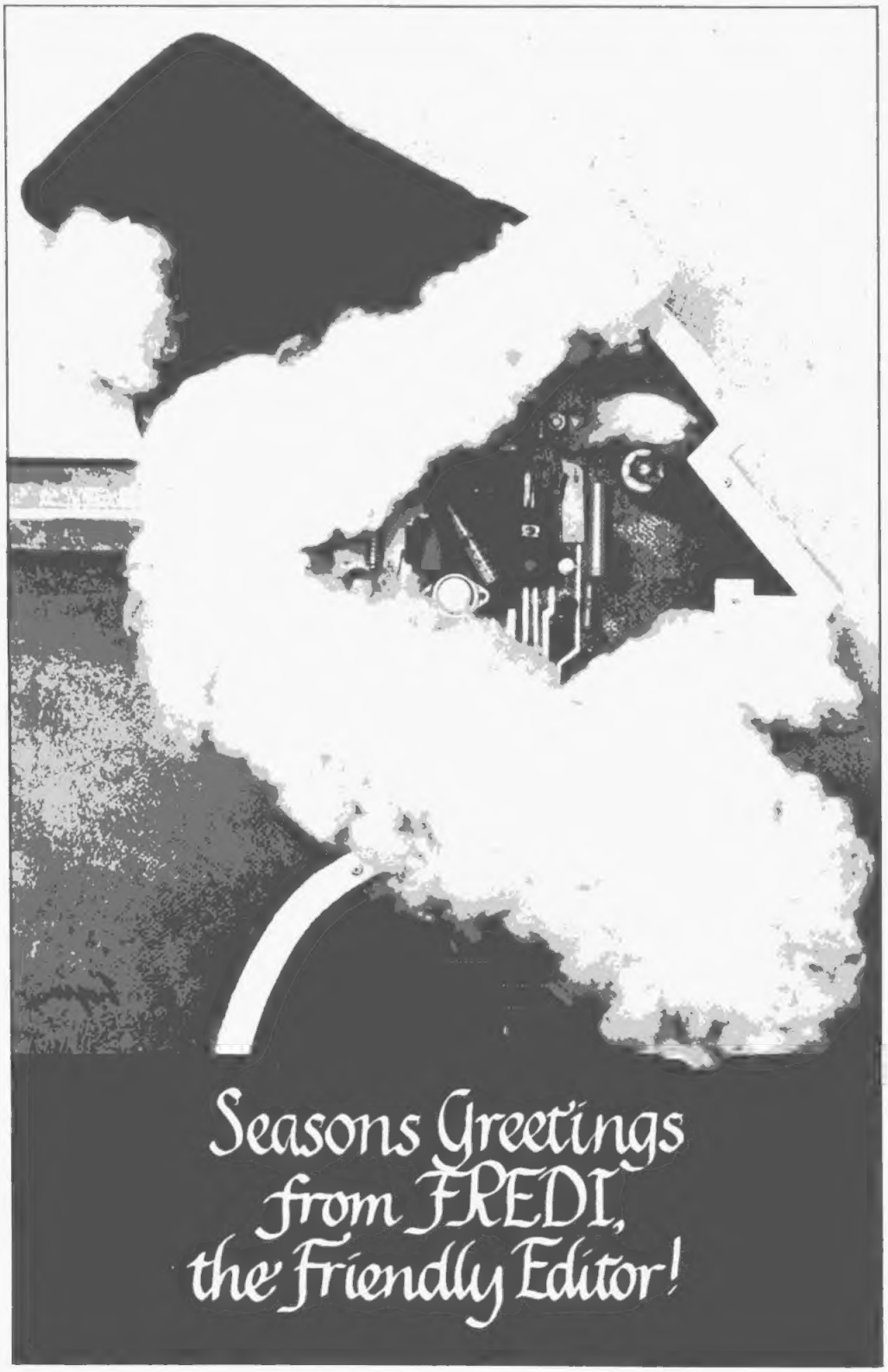
colorcue

a publication for Compucolor users · v.2,# 8 · november 1979 · $1



inside

- SPOTLIGHT ON FREDI
- DOLLARS & CENTS
- ASSEMBLY LANGUAGE 2: THE CHARACTER INPUT ROUTINE
- AND MUCH MORE!

colorcue

menu

3..... EDITOR'S LETTER

REM: SPOTLIGHT ON FREDI

3..... MY FRIEND FREDI, THE BASIC EDITOR

4..... FREDI: SPECIAL APPLICATION

KEEPING IT SIMPLE

6..... DOLLARS & CENTS

7..... LARGE AND SMALL SIZE CHARACTERS

ADVANCED APPLICATIONS

ASSEMBLY LANGUAGE PROGRAMMING 2:

9..... THE CHARACTER INPUT ROUTINE

PRODUCT SHOWCASE

13..... KEYBOARD UPGRADE KITS

OMNI CONTROLS INTRODUCES THE CX10 REMOTE CONTROL DEVICE WHICH MAKES

14..... SECURITY/ENERGY MANAGEMENT A SNAP

MICRO DATA BASE SYSTEMS ANNOUNCES

14..... MDB SYSTEM FOR MICROS

15..... ATTN/BREAK

ENCLOSURES

PRICE LIST AND ORDER FORM

BACK ISSUE ORDER FORM

contributing to the success of this issue

USER INPUT

Bill Greene
F. Lee Newcombe

TECHNICAL ADVICE

Stan Jessen
Knox Pannill

ART DIRECTOR

Debbie Feldman

EDITOR

Cathy Abramson

member international association of business communicators

COLORCUE is published monthly for the users of the Compucolor II personal computer by Compucolor Corporation, P.O. Box 569, Norcross, Georgia 30071. Compucolor is a wholly owned subsidiary of Intelligent Systems Corp. Address all COLORCUE correspondence to our editorial office located at 100 Northcreek, Suite #250, 3715 Northside Parkway, N.W., Atlanta, Georgia 30327, (404) 261-3003. Subscriptions to COLORCUE are $12.00 per year in the U.S., Canada, and Mexico, and $24.00 elsewhere.

editor's letter

This month, our spotlight is on FREDI with an article by Bill Greene describing his first experience with FREDI, the friendly Editor. Bill was editing a large program when he purchased FREDI and because of the size of his program (a version of the French card game Mille bornes) and the memory requirements of FREDI, he ran into difficulties running both together. Bill's introduction to FREDI was a tough test of his ingenuity and persistence but we can all learn from his experience. Read "My Friend FREDI" to find out how he solved his problem. The second article on FREDI clarifies the documentation by telling you how to embed control codes, using the backspace as an example.

Lee Newcombe sent us a subroutine which makes dollar and cent figures appear in proper columns. Lee's subroutine will help make programs with dollar figures much more readable. More is here on Assembly Language Programming for the Compucolor II, with an article on the Character Input routine. We also have a program which allows you to form large and small size characters.

Have you considered upgrading your keyboard lately? An extended or deluxe keyboard will significantly ease Compucolor programming--and will more than pay for itself in convenience and time saving. They're less expensive than you might think--see the article on keyboard upgrades for more details.

The holidays are here and all of us here at Compucolor wish you and your family a delightful and colorful season and with that goes my personal best wishes for the New Year.

=

rem

MY FRIEND FREDI, THE BASIC EDITOR

By Bill Greene
of Byron, Georgia

Initially, FREDI and I were not getting along too well, but it did not take too long to discover that the problem was with me and not FREDI.

When I purchased FREDI, I was working on a large program and was on the verge of going OM (out of memory) on my Model 4. I was attracted by the BASSRC Utility Program and had visions of converting the program to an SRC File and doing some extensive editing, global search and substitution, etc., using my SRC text editor. I soon realized that I had no way of converting the program back to a BAS file. Then I decided to load up FREDI and go the line by line delete and insert route. After loading FREDI, I attempted to load my program and went OM. I learned this time that FREDI occupies about 4K in high memory, meaning that I had to reduce the size of my program somehow. Not giving up easily, and reading about Merge, I decided to convert a rather large instruction section of the main program to a separate program. This freed enough memory for me to load FREDI and edit the remainder of the program. The instruction program was saved for subsequent merging with the main program.

At this point, I had my main program loaded along with FREDI and started editing. Right away, I learned that you can't use the command keys to enter BASIC commands with one stroke line like you can in BASIC (See Editor's Note below). Eager to test the routines that I had modified, I hit "B", Back to BASIC, typed in run, and went OM again. I found that I had enough memory to load my program, but not enough to run it with FREDI sitting up there. Thinking over the situation, I knew that I could save the program, reinitialize BASIC, reload the program and run it. Then I had another idea. Why not just make line 1 of the program, POKE 32940,255:POKE 32941,191? This tells BASIC that FREDI is no longer there and the program will run with full memory available. To do additional editing, it is only necessary to run FRED16 again and edit. This eliminates a lot of unnecessary saving and reloading of the program. The program need only be saved when the editing session is over.

Now FREDI and I are getting along fine, and FREDI is fast becoming my best friend. This diskette has more than paid for itself on this one program in the amount of time that it has saved me.

The MERGE worked beautifully in putting my program back together and my renumbered program certainly appears more professional.

I realize that some of the problems encountered would not have occurred if I had 32K of RAM, but the program could have been a 32K program. The same techniques can be used in either case, except that the POKES for 32K are 255 and 255. However, I am looking forward to working on a smaller program with FREDI standing by, waiting for my call for assistance.

=

EDITOR'S NOTE

That's a great article, Bill, thanks. Let me quickly clear up some confusion. You can enter BASIC commands with one keystroke but, when you do, a single graphic symbol is displayed representing the command. List the new line again, and these symbols are translated into the more familiar written command statement.

=

FREDI -- SPECIAL APPLICATION

The BASIC Editing disk has only been out for a few months and has already become our most popular disk. FREDI's software more than makes up for the lack of a direct editing feature in the Compucolor BASIC.

FREDI is more powerful than many of you may realize. Here's an interesting aspect of FREDI, which allows you to do things not normally possible in BASIC. It involves embedding

control characters in BASIC strings. We can already embed many control characters such as color and flags, for instance. We cannot, however, embed a backspace since BASIC treats this as a signal to erase the last character.

When we enter FREDI and enter or edit a line, the backspace key moves the cursor back one space. How can we enter the control code for a backspace? The answer lies in the BASIC Editing manual but is not very clear. It is accomplished by typing a control-null (both keys depressed at the same time) followed by the desired control key. A backspace is created by entering a control-null followed by the backspace arrow. On the screen it will show up in red and resembles (but is not) the number seven. FREDI displays all control characters in these strange red graphic characters.

The following examples will give you an idea of how handy this backspace feature can be. Since we can't print the control characters in this article, the following substitutions are made

Line Feed -- (LF)
Backspace -- (BS)
Space -- (SP)

Enter the following

```
PRINT"THIS(LF)(BS)(BS)(BS)(BS) IS (LF)(BS)(BS)A (LF)(BS) TEST"
```

Remember that each (LF) and (BS) must be preceded with a control-null. When this statement is run it will look like this:

```
THIS
IS
A
TEST
```

Prefacing the print statement with a PLOT 3,X,Y allows you to print this anywhere on the screen. Here's another example of this feature.

```
PLOT 3,10,15:INPUT "ENTER NUMBER:(SP)(SP)(SP)(SP)(SP)(SP)(SP)(SP)
(BS)(BS)(BS)(BS)(BS)(BS)(BS)(BS)";I
```

When this line is executed it will print "ENTER NUMBER:" followed by 8 blanks and then back the cursor up 8 spaces. This gives you a simple method of blanking out the previous value of an input statement when you wish to use the input statement at the same location.

Not all control codes can be entered with FREDI. For example, attempting to embed an "erase page" causes FREDI to bomb. We can confirm that all four cursor commands will work. This same technique of using the control-null is used when "[" or "]" is needed.

With a little imagination, you'll find many other unique things that FREDI can do for you. For those of you who don't have it yet, run down to your local computer store or write to us for more information. Don't miss out on one of Compucolor's most powerful programs.

=

keeping it simple

DOLLARS & CENTS

By F. Lee Newcombe of Zephyrhills, Florida

I have found that most of my programs involve accounting for money and result in displaying dollar and cents amounts. The program which keeps my checkbook is one example. Unfortunately, if you print the numeric amounts, the display looks like this:

```
AMOUNT 1.   $102.35
AMOUNT 2.   $.55
AMOUNT 3.   $10.5

TOTAL       $113.4
```

To line it all up for a better appearance and to make it easier to read, I wrote the following subroutine. I use large line numbers so I can use the same line numbers in all of my programs.

FORMAT MONEY AMOUNTS

```
6000 REM N=NUMERIC VALUE TO BE FORMATTED
6010 REM N$=STRING VALUE TO PRINT MONEY AMOUNT
6020 IN=INT(SGN(N)*N)
6030 RE=(SGN(N)*N)-IN
6040 N$="   "+STR$(SGN(N)*IN): REM "   " CONTAINS 3 SPACES
6050 IF RE=0 THEN N$=N$+".00":GOTO 6070
6060 N$=N$+MID$(STR$(RE+.001),2,3)
6070 N$="$"+RIGHT$(N$,7)
6080 RETURN
```

Here's how it works:

1. Transfer your numeric value to the symbol "N" before going sub.

```
100  X=15
105  N=X
110  GOSUB 6000
```

2. Determine the positive whole number contained in "N" (Line 6020).

3. Determine the positive decimal number contained in "N" (Line 6030).

4. Start string with 3 blanks and add sign of "N" plus integer (Line 6040).

5. If decimal amount is "N"=0 then add ".00" to string N$.

6. If "N" contains a decimal amount then add that amount to the string N$ (Line 6060).

7. Set N$ length to 8 spaces with "$" first.

Finally, you can print the dollar and cents amount as a string value with the statement:

```
115  PRINT N$
```

=

If you would like to expand this program to accomodate larger numbers, make the following changes: **EDITOR'S NOTE**

```
6040 N$="6sp"+LEFT$(STR$(SGN(N)),1)+MID$(STR$(N),2):
     REM "    "CONTAINS 6 SPACES
6070 N$="$"+RIGHT$(N$,10)
```

This needs to be done to avoid printing -.01 as +.01. Line 6040 also allows you to start the string with 6 blanks instead of 3. Line 6070 allows you to set N$ length to 11 spaces with "$" first.

For sophisticated programmers, this two line program does the same job. Can you determine its functions?

```
60000 N$=LEFT$(STR$(SGN(N)),1)+MID$(STR$(INT(ABS(N))),2)
      +MID$(STR$((ABS(N))-INT(ABS(N))+1.001),3,3)
60010 N$=RIGHT$("6sp$"+N$,11):RETURN
```

=

LARGE SPECIAL CHARACTERS

Here's a simple BASIC routine which will take a string of characters and print them out in large special characters. The program will also accept control characters such as color, blink, and large and small character sizes (A7 ON and A7 OFF). The special characters that make up each large character are saved in an array, and are referred to each time they are needed.

At the beginning of the program, memory space is set aside for the array LE. LE is then dimensioned for 59 sets of 4 characters [LE (59,4)] and now the data can be read into the array. The data itself is taken almost directly from the CCII **Programming Manual.** When the subroutine at line 500 is used, the Input A$ is taken and printed character by character through the use of the MID$ function. If the character is a

control character (ASCII value less than 32), it is plotted normally. If its value is greater than 90, it is ignored. If the character is between 32 and 90, it is printed from the values in the array.

Because some characters require the flag to be on (columns 6 & 7 in Appendix E) and others require the flag to be off (columns 1 & 2), the flag is set as the character is printed. For example, the letter K requires the special characters 98, 8, 1, and 28. When the value is taken from the array, the character value 29 (for FLAG OFF) is plotted, followed by the 98 (the character from the array). For the value 8, a 30 (for FLAG ON) is plotted. Next, the character is printed with a value of 96 added to it. This is done to get the proper special characters, otherwise control characters would be plotted.

```
100 DIM LE(59,4):REM SET UP ARRAY SPACE
110 POKE 33289,255:REM SET LENGTH OF LINE AT MAXIMUM
115 REM CREATE CHARACTERS TO CONTROL THE CURSOR
120 SP$(1)= CHR$ (10)+ CHR$ (26):REM GOES DOWN AND BACK
130 SP$(3)= CHR$ (28)+ CHR$ (26):REM GOES UP AND BACK
135 SP$(4)= CHR$ (32):REM             SPACES
140 FOR X= 1TO 59:FOR Y= 1TO 4
150 READ LE(X,Y):REM READ VALUE INTO ARRAY
160 NEXT Y,X
170 REM
180 REM
500 PLOT 6,2,15,29:PRINT :PRINT :PRINT
510 PRINT "PLEASE ENTER YOUR STRING: ":INPUT A$:GOSUB 700
515 IF LEN (C$)= 0THEN 500
520 FOR Y= 1TO LEN (A$)
530 C= ASC (MID$ (A$,Y,1)):REM GET CHARACTER'S ASCII VALUE
540 IF C> 90THEN 600:REM SKIP INVALID CHARACTERS
550  IF C< 32THEN PLOT 29,C:GOTO 630:REM PLOT A CONTROL
    CHARACTER
560 M= PEEK (33227):REM GET X CURSOR POSITION
570 X= C- 31:REM GET SUBSCRIPT OF CHARACTER IN ARRAY
575 REM LINE 580 - IF AT END OF LINE SET UP NEXT LINE
580 IF M> 60THEN GOSUB 700:PRINT :PRINT
590 FOR Z= 1TO 4
600 IF LE(X,Z)= > 32THEN PLOT 29:PRINT CHR$ (LE(X,Z));
610 IF LE(X,Z)< 32THEN PLOT 30:PRINT CHR$ (LE(X,Z)+ 96);
620 PRINT SP$(Z);:NEXT Z
630 NEXT Y:GOTO 500
699 REM THESE LINES SET UP LINE TO PRINT ON
700 PLOT 6,2,15,29
710 PLOT 10,10,10,10
720 PLOT 28,28,28,28:RETURN
9999 REM DATA FOR ARRAY ON EACH CHARACTER
10000 DATA 32,32,32,32, 32,32,33,110, 2,32,32,2, 43,43,43,43,
    3,13,4,14
```

```
10010 DATA 79,30,79,30, 3,3,24,4, 32,32,32,30, 20,22,32,32,
   32,32,23,21
10020 DATA 19,26,16,25, 2,15,5,32, 32,30,32,32, 32,5,5,32,
   32,42,32,32
10030 DATA 32,30,32,30, 0,22,24,21, 8,32,1,1, 8,25,31,4,
   5,7,4,26
10040 DATA 30,5,14,1, 27,7,4,5, 20,123,4,6, 5,30,32,26,
   3,3,4,4
10050 DATA 3,7,23,125, 42,42,32,32, 42,30,32,32, 6,9,8,7,
   105,102,102,105
10060 DATA 9,6,7,8, 8,32,1,4, 20,22,9,4, 30,14,15,28,
   27,12,4,4
10070 DATA 20,22,9,6, 14,12,23,21, 27,12,31,5, 27,1,32,5,
   20,22,29,6
10080 DATA 1,14,15,2, 5,31,12,14, 32,7,23,2, 98,1,28,8,
   1,12,31,32
10090 DATA 16,1,2,17, 16,1,19,2, 20,22,23,21, 27,1,32,4,
   20,22,24,21
10100 DATA 27,1,28,4, 3,7,4,6, 5,32,1,14, 1,22,23,2, 1,28,30,2
10110 DATA 1,18,19,2, 28,30,28,30, 28,2,32,30, 5,25,31,26
```

=

advanced applications

ASSEMBLY LANGUAGE PROGRAMMING -- PART 2

In part 1 of this series, we talked about the 'LO' and 'OSTR' routines. Now we will discuss the 'CI' or character input routine. This routine is not located in the system software and for this reason we must concern ourselves not only with the routine itself, but also with the initialization and interrupt routines that go with it.

Addresses for routines referenced in this article are listed below.

```
INPCRT  EQU     81C5H   ; JUMP VECTOR #31
KBDFL   EQU     81DFH   ; HOLDS NUMBER OF JUMP
                        ;  VECTOR FOR THE
                        ;  KEYBOARD
KBRDY   EQU     81FFH   ; KEYBOARD READY FLAG
```

The addresses above are the same for both V6.78 and V8.79 system software.

The first routine needed for character input is the initialization routine, 'CIINIT'. It should be placed at the beginning of the program before any I/O takes place. The routine is as follows:

```
;;        CIINIT - The Character Input Initialization
;                  routine sets up the parameters necessary
;                  for the 'CHRINT' and 'CI' routines.
;
CIINIT:   MVI     A,31     ; SETUP JUMP VECTOR #31
          STA     KBDFL    ; STORE IN KEYBOARD FLAG
          MVI     A,0C3H   ; LITERALLY 'JMP'
          STA     INPCRT
          LXI     H,CHRINT ; GET ADDRESS OF 'CHRINT'
                           ;       ROUTINE
          SHLD    INPCRT+1 ; PLACE ADDRESS AFTER 'JMP'
          XRA     A
          STA     KBRDY    ; CLEAR KEYBOARD READY FLAG
          STA     CHARIN   ; CLEAR TEMPORARY CHARACTER
                           ; STORAGE
;
```

The 'CIINIT' routine first sets the vector number for the keyboard interrupt and places it in the keyboard flag, 'KBDFL'. When a 'JMP' and the address of the character interrupt routine, 'CHRINT', are placed in the #31 vector storage location, 'INPCRT', the keyboard ready flag is cleared along with the temporary character storage location. It should be noted that 'CHARIN' is a location defined by the user.

```
CHARIN:   DS      1        ; TEMPORARY CHARACTER STORAGE
```

The interrupt routine gets a character and stores it for the 'CI' routine.

```
;;        CHRINT - The Character Interrupt routine is
;                   vectored to from the keyboard input
;                   routine through the jump vector
;                   'INPCRT' (#31). The character from the
;                   keyboard input routine is in
;                   register E.
;
CHRINT:   PUSH    H        ; SAVE REGISTERS THAT WILL BE
                           ; USED
          PUSH    PSW
          LXI     H,CHARIN ; GET ADDRESS OF TEMP
                           ; CHARACTER STORAGE
          XRA     A        ; CLEAR A
          CMP     M        ; TEST FOR 'CHARIN' = 0
          JNZ     CFIN     ; IF NOT ZERO, THEN IGNORE
                           ; INPUT
          MOV     A,E      ; GET CHARACTER FROM E
          ANI     127      ; STRIP UPPER BIT FOR ASCII
          MOV     M,A      ; STORE IN 'CHARIN'
CFIN:     POP     PSW      ; RESTORE USED REGISTERS
          POP     H
          EI               ; ENABLE INTERRUPTS
          RET              ; RETURN FROM INTERRUPT
;
```

'CHRINT' first saves the registers that will be used during the interrupt service. Next the temporary storage

location, 'CHARIN', is checked for a byte value of zero. If the value equals zero, the new character is saved, if not, the new character is ignored. The upper bit is stripped off for ASCII. The registers are restored and the interrupts are enabled for the next keyboard interrupt. The keyboard is scanned approximately 60 times a second and the keyboard interrupt is generated by an interrupt timer.

The following character input routine does not allow the buffering of more than one character. When 'CI' is called, the character taken will be the first character keyed after the last character that was taken. Buffered input is possible with some modifications to this routine and to 'CHRINT'. A multi-character buffer may be a topic for a later article.

```
;;      CI - The Character Input routine gets a
;               character from the temporary storage
;               location 'CHARIN', clears the keyboard
;               ready flag and returns with the
;               character in A.  If there is no
;               character in 'CHARIN', then 'CI' will
;               hang and wait for a character.
;
CI:     EI              ; ENABLE INTERRUPTS
        LDA     CHARIN  ; GET CHARACTER
        CPI     0       ; HAVE A CHARACTER ?
        JZ      CI      ; IF NOT, HANG FOR CHARACTER
        PUSH    PSW     ; SAVE CHARACTER
        XRA     A
        STA     KBRDY   ; CLEAR KEYBOARD READY FLAG
        STA     CHARIN  ; CLEAR TEMPORARY STORAGE FOR
                        ;  NEXT CHAR
        POP     PSW     ; RESTORE CHARACTER
        RET             ; ECHO CHARACTER FROM MAIN
                        ;  ROUTINE
;
```

The 'CI' routine first checks for the presence of a new character in 'CHARIN'. If there isn't a character, the 'CI' waits for a character to be struck. Once a character is present, it is saved, and the keyboard ready flag and temporary storage location are cleared for the next input.

At this point it is advantageous to echo the character to the screen. If the echoing, or call to 'LO', is performed in a 'main' user input routine, some kind of checking can be performed on the character to make sure that they will not cause an undesired result. This also allows you to translate any characters for specific tasks. An example might be to translate the 'back arrow' key into a backspace, space, backspace which will backup the cursor and delete the last echoed character from the screen. The section of the main input routine may look like this (the last character from input is assumed to be in A):

```
        CALL    CI      ; GET A CHARACTER
         .
         .
         .
TBS:    CPI     26      ; CHECK FOR LEFT ARROW (BCKSPC)
        JNZ     NTEST   ; IF NOT, GO TO NEXT TEST
        MVI     A,26
        CALL    LO      ; SEND BACKSPACE TO SCREEN
        MVI     A,' '
        CALL    LO      ; SEND SPACE
        MVI     A,26
        CALL    LO      ; SEND ANOTHER BACKSPACE
        JMP     NCHAR   ; GO GET NEXT CHARACTER

NTEST:  ; next test for specific character(s)
```

Here are some related addresses. They will be described in future issues but are being included here for reference purposes.

```
KBCHA   EQU     81FEH   ; CHARACTER FROM
                        ;  KEYBOARD
KEYTST  EQU     0024H   ; KEYBOARD SCANNER
```

The following equivalents may help your program be more readable.

```
CTRLB   EQU     02      ; CONTROL B
CTRLC   EQU     03      ; CONTROL C
BELL    EQU     07      ; BELL
HT      EQU     09      ; HORIZONTAL TAB
LF      EQU     10      ; LINEFEED
VT      EQU     11      ; VERTICAL TAB
                        ;  (OR ERASE LINE)
FF      EQU     12      ; FORM FEED
                        ;  (OR ERASE PAGE)
CR      EQU     13      ; CARRIAGE RETURN
```

The character input routines allow you to take input from the keyboard. The routines are best left general purpose while the routines that call 'CI' can be written for a particular purpose or application. Again it is recommended that you retain the labels given for the variables so that any questions you may have at a later time will be easier to answer.

Send us your suggestions for future articles on Assembly. Any comments or reference recommendations would also be appreciated. Next month, we will look at a routine to make the software you write compatible with both V6.78 and V8.79 software.

product showcase

KEYBOARD UPGRADE KITS

If you are as lazy as I am, keyboard upgrade kits are good news! You can add a lot of convenience and save yourself time and labor by adding a more versatile keyboard to your Compucolor.

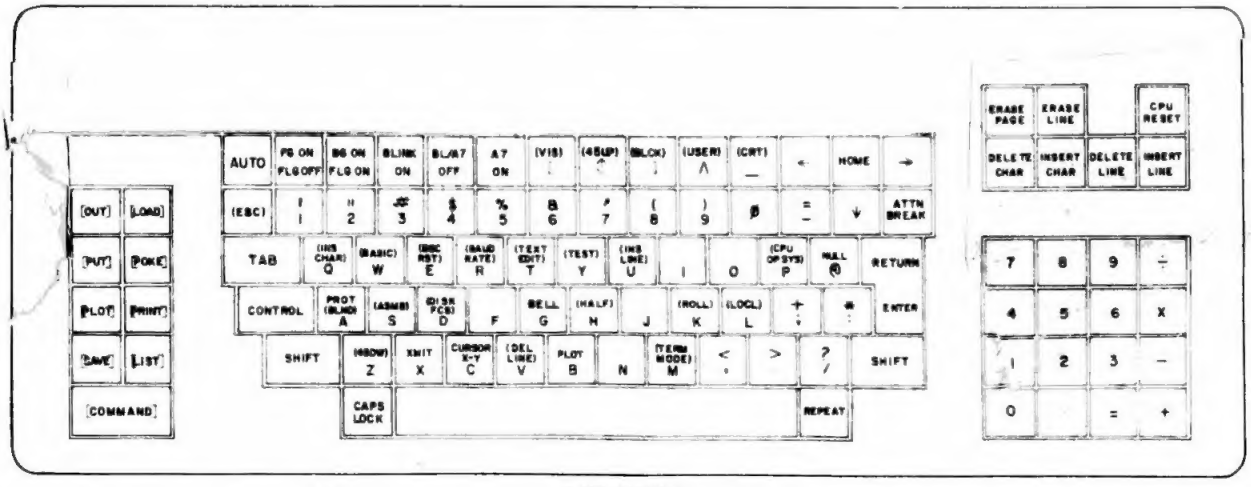
When you upgrade from the 72-key to the extended 101-key keyboard, you add a special command pad which allows you to enter eight of the most common BASIC commands with one keystroke. This pad also lets you select colors with ease. In addition, the 101-key keyboard includes a separate number pad that makes rapid number entry a breeze.

The deluxe 117-key keyboard expands flexibility to the limit by including 16 special-function keys to simplify plotting, vectoring, and graphing. These keys also give you an additional 64 ASCII codes. Note: the deluxe 117-key keyboard is required to run the Screen Editor Sof-Disk.

Compucolor has made it easy to upgrade your keyboard inexpensively. Upgrade kits are available at the following prices:

72 key to 101 key	$150
72 key to 117 key	$215
101 key to 117 key	$100

The kits are readily available and can be installed using a soldering gun. If you don't care to install it yourself, your Compucolor dealer can install it for you or we will upgrade your keyboard at the factory for a $25 installation fee, shipping, plus the price of the kit. If you don't know the Compucolor dealer in your area, call us at (404) 449-5996 for the one closest to you.



117 KEYS

=

As a service to our readers, Compucolor Corporation publishes information sent to us by other manufacturers of products, techniques, or scientific and technological developments which can be used in conjunction with the Compucolor computer. However, in many cases, Compucolor Corporation has not had the opportunity to thoroughly examine or test these products. Therefore, we neither endorse nor assume any responsibility or liability for the proper functioning of products not directly manufacturered by the Compucolor Corporation. We hope you will find these products of interest, but we encourage you to investigate carefully before you buy.

=

OMNI CONTROLS INTRODUCES THE CX10 REMOTE CONTROL DEVICE WHICH MAKES SECURITY/ENERGY MANAGEMENT A SNAP

The cordless Model CX10 remote control device, available from Omni Controls, allows the Compucolor II to control temperature and turn lights and other appliances on or off on a preprogrammed schedule while you sleep or are away from your home or office. In addition, the CX10 system includes a wireless remote data entry keyboard so that you can use your Compucolor from as far away as 30 feet.

No modifications to your CCII are necessary. The CX10 works on regular AC power lines. Omni Controls is offering the CX10 at the special introductory price of $275. Contact them directly for further details.

Omni Controls
P.O. Box 48541
Atlanta, Georgia 30362
(404) 581-0243

=

MICRO DATA BASE SYSTEMS ANNOUNCES MDB SYSTEM FOR MICROS

```
CR      EQU     13      ; CARRIAGE RETURN
```

ement sys $750. The system is available in 8080 code and requires 8-16K b[illegible]s. This system includes a Data Definition Language Analyzer/Editor, Data Manipulation Language, a user's manual and sample application programs. Micro Data Base Systems anticipates a price increase after the first of the year, so if you are interested, contact them immediately. They can be reached at:

Micro Data Base Systems, Inc.
P.O. Box 248
Lafayette, In. 47902
(317) 742-7388

=

attn/break

The last program line on page 13 of the September/October issue of **COLORCUE**, Keeping It Simple: Random Files 2 should read:

```
700 GET 1,RC;M$[8],P,M,MA,E
```

=

Recently, a number of people have asked how to print out the Directory.

In order to print out the Directory, you **must** be using a serial printer. The code is:

(Escape) G DIR (Return)

=

Happy New Year from Compucolor!

Compucolor® Corporation

Intecolor Drive • Technology Park/Atlanta • Norcross, Georgia 30092 • Telephone 404/449-5996